52-16-00   A

jc759 U.S. PTO
02/14/00

NEW UTILITY PATENT APPLICATION TRANSMITTAL

(only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket Number	4464
First Named Inventor	Pawan Goyal
Total Pages in this Submission	88
Express Mail Label No.	EM083021125US

09/503975   U.S. PTO
02/14/00

APPLICATION ELEMENTS	ACCOMPANYING APPLICATION PARTS

APPLICATION ELEMENTS

1. ☑ Fee Transmittal Form (in duplicate)

    ☑ Check Enclosed

2. ☑ Specification
   (preferred arrangement set forth below)
   - Descriptive Title of the Invention
   - Cross Reference(s) to Related Case(s)
   - Statement Regarding Fed sponsored R & D
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawing(s)
   - Detailed Description
   - Claim or Claims
   - Abstract of the Disclosure

3. ☑ Drawing(s) *(when necessary per 35 USC 113)*

4. Oath or Declaration

   a. ☑ New Declaration

      ☑ Executed

   b. ☐ Copy from a prior application (37 CFR 1.63(d))
      (for continuation/divisional with Box 17 completed)

      i. ☐ DELETION OF INVENTOR(S)
         Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. ☐ Incorporation by Reference *(useable if Box 4b is checked).* The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

ACCOMPANYING APPLICATION PARTS

6. ☐ Assignment & Assignment Recordation Cover Sheet

7. ☐ Certified Copy of Priority Document(s)
   (if foreign priority is claimed)

8. ☐ Information Disclosure Statement & PTO-1449

      ☐ Copies of IDS Citation(s)

9. ☐ Preliminary Amendment

10. Small Entity Statement

      ☑ New Statement enclosed

      ☐ Statement filed in prior application. Status still proper and desired

11. ☑ Return Postcard

12. ☐ _____

13. ☐ _____

14. ☐ _____

15. ☐ _____

16. ☐ _____

ADDRESS TO:

Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

17. **If a CONTINUING APPLICATION,** *check appropriate box and supply the requisite information below and in a preliminary amendment:*

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP)   of prior application No: ___/_____

Prior application information: Examiner: _____   *Group/Art Unit:* _____

18. CORRESPONDENCE ADDRESS

NAME	Albert C. Smith Fenwick & West LLP					
ADDRESS	Two Palo Alto Square					
CITY	Palo Alto		STATE	CA	ZIP CODE	94306
COUNTRY	U.S.A.	TELEPHONE	(650) 858-7296		FAX	(650) 494-1417
Name (Print/Type)	Albert C. Smith		Registration No. (Attorney/Agent)			20,355
Signature	*[signature]*			Date		2/14/2000

21816/04464/DOCS/1000626.1

VERIFIED STATEMENT CLAIMING SMALL ENTITY STATUS (37 CFR 1.9(f) & 1.27(c))--SMALL BUSINESS CONCERN	Docket Number (Optional): 4464

Applicant or Patentee: _____ Pawan Goyal _____

Application or Patent No.: _____ not yet known _____

Filing Date or Issue Date: _____ not yet known _____

Title: RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO A SET OF SPECIFIC NETWORK ADDRESSES

I hereby declare that I am

[ ] the owner of the small business concern identified below:

[X] an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF SMALL BUSINESS CONCERN _____ Ensim Corporation _____

ADDRESS OF SMALL BUSINESS CONCERN _____ 1215 Terra Bella Avenue, Mountain View, CA 94043 _____

    I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees to the United States Patent and Trademark Office, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

    I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention described in:

[ ] the specification filed herewith with title as listed above.

[X] the application identified above.

[ ] the patent identified above.

    If the rights held by the above identified small business concern are not exclusive, each individual, concern or organization having rights in the invention must file separate verified statements averring to their status as small entities, and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 37CFR 1.9(c) if that person made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d), or a nonprofit organization under 37 CFR 1.9(e).

    Each such person, concern or organization having any rights in the invention is listed below:

[X] No such person, concern, or organization exists.

[ ] Each such person, concern or organization is listed below:

_____

_____

    Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

    I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

    I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING Srinivasan Keshav

TITLE OF PERSON IF OTHER THAN OWNER Chief Technical Officer

ADDRESS OF PERSON SIGNING 1215 Terra Bella Avenue, Mountain View, CA 94043

SIGNATURE _____ S. K. _____ DATE _____ 2/11/00 _____

21816/04464/DOCS/998971.1

RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO A SET OF SPECIFIC NETWORK ADDRESSES

INVENTOR

Pawan Goyal

Prepared by:

Fenwick & West LLP
Two Palo Alto Square
Palo Alto, CA 94306

Express Mail No.:
EM083021125US

RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO A SET OF SPECIFIC NETWORK ADDRESSES

INVENTOR

Pawan Goyal

BACKGROUND

FIELD OF INVENTION

The present invention relates generally to computer network communication by software processes, and specifically to restricting process communication to a set of specific network addresses.

BACKGROUND OF INVENTION

With the popularity and success of the Internet, server technologies are of great commercial importance today. Typically, a single server program executes on a physical host computer, and services client requests made to the host. Most commonly, one network address is assigned to a physical host. However, using Transmission Control Protocol/Internet Protocol (TCP/IP) and other transport protocols, more than one network address can be assigned to a single physical host computer. Where a single network address is assigned to a physical host, the server program services client requests made to the single network address. Where multiple network addresses are assigned, the server program services client requests made to the multiple network addresses.

To service requests made to a host, a server program executing on the host typically opens a communication transport channel (socket) and allows receipt of incoming communications targeted for any of the network addresses assigned to the host. Accepting a communication request by a server executing TCP/IP is a three-step process that includes waiting for the communication request from a client, sending an acknowledgment signal to the client, and receiving a return acknowledgment signal from the client. This three-step process is called "three way handshaking," and is a feature of TCP/IP communication.

A server program is simply a process. Multitasking operating systems can execute multiple processes simultaneously, so it is technically possible for more than one server program to execute on a single physical host computer. The ability to execute multiple server programs on a single physical host is desirable, because providing a unique physical host for each server program is expensive and inefficient. Hosting services are often provided commercially by an Internet Service Provider (ISP). Absent the execution of multiple server programs on a single physical host, an ISP would have to provide a separate physical host computer for every customer that purchases host services. Often, a customer purchasing host services from an ISP will neither require nor be amenable to paying for use of an entire host computer. Generally, only a fraction of the processing power, storage, and other resources of a host computer will be required to meet the needs of an individual customer.

Execution of multiple server programs on a single host would allow an ISP to utilize one host computer to provide commercial host services to multiple customers. Each customer would be assigned a single server program, and would be provided with

resources on the single, physical host computer, effectively sharing the host with other customers. A client computer would request data from a specific one of the servers by targeting communication requests to one of the network addresses of the host computer. Thus, the functionality of numerous hosts would be provided by a single physical host computer, servicing requests made to a plurality of server programs by multiple customers.

One problem that renders the execution of multiple servers on a single physical host commercially unviable today is the inability to restrict the communication of individual servers to a set of specific network addresses. There are two options by which a server program can register itself with the operating system to receive incoming communication requests. The first option is for a server to register itself to receive communication requests targeted to any of the network addresses of the physical host computer. A server program registered according to the first option receives communication requests arriving at all of the network addresses allocated to the host. Thus, multiple server programs so registered can execute simultaneously and service requests made to the network addresses associated with the physical host, but specific ones of the server programs can not be restricted to receiving and servicing requests made to specific ones of the network addresses allocated to the physical host. Thus, any request made by any client to any network address allocated to the physical host could be received by any one of the server programs executing on the host.

Commercially desirable server programs must be associated with specific network addresses. Each customer of an ISP wants their server to receive and respond only to requests made thereto. Furthermore, each customer wants only their server to receive its

targeted requests. Customers could benefit from the lowered expense of executing

multiple server programs on a single physical host, but of course would insist on privacy

between the multiple servers.

Customers would not accept a system in which a request targeted to their server

5   could be received by a server of another customer of the ISP. The other customer could

be a competitor, and the request could comprise classified data. And of course, servers

are not generally programmed to process requests intended for other servers, and thus

requests received by another server could go unprocessed, or could be improperly

processed. Even in the best case scenario in which a server could process a received

10   request intended for another server, the processing server would be allocating resources,

paid for by one customer of the ISP, to service a request made to another customer.

Clearly, the execution on a single host of multiple server programs which are not

associated with specific network addresses is totally unacceptable for commercial

purposes.

15   The second option by which a server program can register itself with the

operating system to receive incoming communication requests is for the process to

register itself to receive communication requests targeted to an individual network

address associated with the physical host. Server programs registered according to the

second option receive communication requests arriving only at a specific individual

20   network address of the host. Thus, multiple server programs so registered can execute

simultaneously. Each server program receives and services requests made to a specific

one of the network addresses associated with the physical host. However, each server is

restricted to receiving and servicing requests made to only one of the network addresses

allocated to the physical host. Thus, no server program can service requests made to multiple network addresses.

While it is desirable to be able to restrict a server program to communication via a specific set of network addresses, it is at the same time desirable to be able to include in the specific set more than one network address. Many existing server programs that execute on dedicated physicals hosts are configured to service communication requests made to multiple network addresses. This functionality is a popular feature with purchasers of commercial host services. Were an ISP to commercially offer multiple server programs executing on a single physical host computer, the customers would expect the servers to be able to communicate via multiple network addresses. Thus, it is desirable for an ISP to be able to provide, on a single physical host computer, multiple servers each of which can service requests made to multiple network address.

There is an additional security related problem that results from not being able to restrict server programs to communication via specific network addresses. Without a mechanism to restrict a process to accessing a specific set of network addresses, a server program could be written that intentionally receives or monitors communication requests made to another server executing on the same host. If a customer of an ISP or an unauthorized third party learned a network address associated with another customer's server, it would be possible for the unauthorized party to create a server program to receive or monitor, at a source code level, communication via that address. The risk of such activity would obviously be unacceptable to customers of ISP's. Of course, ISP's could examine the source code of all server programs to attempt to prevent such activity, but such checking would be time consuming and expensive. It would be desirable for a

process executing on the host, external to every server program, to ensure that no server program communicate via any unauthorized network address.

It is also important to understand that many server programs are being provided today by ISP's and other providers of host services. As explained above, it would be desirable for existing ISP's to be able to provide multiple server programs on a single physical host. However, many such ISP's would not want to replace their existing server programs with ones that could overcome the problems associated with providing multiple servers on a single physical host, even if such servers were available. Upgrading server software is a time consuming and complicated process, often involving costly down time and high labor expenses. It would be desirable to have a system to allow existing providers of server programs to provide multiple server programs on a single physical host without having to upgrade or replace their existing server software.

In summary, what is needed is a method whereby a process can be restricted to communication via a set of specific, multiple network addresses. That way, ISP's could provide multiple, commercially viable server programs on a single physical host computer. Furthermore, the method should be external to server programs executing on a host, so that unauthorized servers and third parties can be prevented from monitoring communication of other server programs. Finally, the method should not require the replacement of existing server programs.

SUMMARY OF INVENTION

The present invention allows the restriction of process communication to a specific set of network addresses. In order to facilitate such restriction, selected processes are associated with specific network addresses. Network address-based

5   communication of the selected processes is restricted to the associated network addresses. Certain attempts by selected processes to facilitate or conduct communication via a network address are detected, including attempts to designate a network address to be used for subsequent communication. Only in response to a determination that the network address is associated with the process is the designation, and hence the

10   subsequent communication, allowed to proceed. Also detected are attempts by selected processes to communicate without first designating a specific network address. Before such communication is allowed to proceed, an associated network address is designated for the communication.

Selected processes that are to be restricted to communication via a set of specific

15   network address are loaded by a modified loader program. A loader program is an operating system utility that is used to execute computer programs that are stored on static media. Typically, a loader program loads an executable image from static media into process address space, and then initiates execution of the loaded image by transferring execution to the first instruction thereof.

20   Like a standard loader program, the modified loader of the present invention loads executable images from static media into process address space. Additionally, the modified loader associates each loaded processes with a specific set of network addresses. The set can comprise one or more network address. The loader program

associates the process with the set of addresses by storing an association between a process identifier of the process and the set of network addresses. In one embodiment, the associations are stored in an association table. In other embodiments, the associations are stored in other data structures as desired, for example a linked list. The association

5   table (or other data structure) is preferably stored in operating system address space, but can also be stored in user space as desired. Each process that is to be restricted to communication via a specific set of network addresses is loaded by the modified loader program in this manner.

The loader program also associates the process with a dedicated local host

10   address. A local host address is an address that is used to conduct intra-computer communications between a server program and other processes running on the same physical computer. On a dedicated physical host computer, only one local host address is necessary because only a single server program executes on that computer. All intra-computer communication with the server program is conducted via the single local host

15   address. In TCP/IP, the single local host address is 127.0.0.1. On every physical host running TCP/IP, this address is reserved for intra-computer TCP/IP communication. However, where multiple server programs (processes) execute on a single physical computer (or in any scenario in which intra-computer communication can be targeted to a plurality of processes), each such process must have its own local host address via which

20   it can conduct intra-computer communication. If each server program utilized the same local host address, intra-computer communication targeted for any server program could be received by any other server program.

In order to ensure that all intra-computer communication is private, whenever the modified loader program loads a selected process, the modified loader associates the selected process with a dedicated local host address. This association is then stored by the loader program. The dedicated local host address is reserved for the exclusive

5   utilization of the selected process. The process uses its dedicated local host address for the sole purpose of intra-computer communication. Once the associations have been stored, the loader program proceeds to initiate execution of the loaded process by transferring execution to the first instruction thereof.

The stored associations are used to restrict the communication of processes to

10   associated network addresses. In order to keep the associations complete and current, the present invention intercepts system calls that create child processes, and system calls that delete processes. System calls that create child processes are intercepted in order to associate a child process with the network addresses of the parent process. System calls that delete processes are intercepted in order to delete an association between a deleted

15   process and a set of network addresses.

When a system call that creates a child process is intercepted, an interception module examines the association table (or other data structure in which associations are stored) to determine if the parent process is associated with a set of network addresses. If so, the interception module allows the creation of the child process to proceed, and then

20   updates the association table to include an association between the child process and the set of network addresses with which the parent process is associated. Communications of the child process are restricted to this set of network addresses. If the parent process is

not associated with a set of network addresses, the interception module simply allows the creation of the child process to proceed normally.

When a system call that deletes a process is intercepted, the interception module examines the association table to determine if the process to be deleted is associated with a set of network addresses. If so, the interception module allows the deletion of the process to proceed, and then deletes the association in the table between the deleted process and the set of network addresses. If the process is not associated with a set of network addresses, the interception module allows the deletion of the process to proceed normally.

The present invention detects when a selected process is attempting to designate a network address for subsequent communication, and when a selected process is attempting to conduct network-address based communication that does not require the pre-designation of a network address. In order to so detect, preferably the Transmission Control Protocol/Internet Protocol (TCP/IP) stack of the operating system is modified so as to intercept certain communication protocol subroutines that are associated with network address-based communication. Modifying a communication protocol stack to intercept subroutines is similar to intercepting system calls. A communication protocol stack includes pointers to subroutines that perform various communication based tasks. The protocol stack is modified by replacing the pointers to certain subroutines that pertain to network address-based communication with pointers to the interception module, such that when the subroutines are called, the interception module executes instead. Examples of network address-based communication tasks performed by communication protocol stack subroutines include creating a communication channel,

associating a communication channel with a network address, receiving data via a communication channel, and transmitting data via a communication channel.

When a network address-based communication subroutine is intercepted, the interception module examines the association table to determine whether the process that called the subroutine is associated with a set of network addresses (in other words, the interception module determines whether the process is selected). If a selected process is attempting to designate a network address for subsequent communication, the interception module ensures that the network address is associated with the process. If the network address is not associated with the process, the interception module generates an error condition, and does not allow the designation or subsequent communication to proceed. If a selected process is attempting to engage in network address based communication without designating a network address to be utilized for the communication, the interception module ensures that the communication is conducted via an associated address.

If a selected process is attempting to conduct network-address based communication that does not require the pre-designation of a network address, the interception module determines if a network address has been pre-designated. If not, the interception module designates an associated address. If an unassociated address has been designated, the interception module does not allow the communication to proceed, and instead generates an error condition. Thus, processes that are associated with a set of network addresses are restricted to executing network based communication via the associated set of addresses. Where the process that called the subroutine is not associated

with a set of network addresses, the interception module allows the communication to proceed normally.

In an alternative embodiment of the present invention, rather than modifying a communications protocol stack, the present invention intercepts system calls that pertain to network address-based communication. Interception of these system calls achieves the same result as the modification of the communication protocol stack. A communication protocol stack is at a lower level than a system call. In fact, in order to request a communication task, a process typically makes a system call. The system call, in turn, calls the subroutine in the appropriate communication protocol stack. Thus, intercepting either the system call (higher level) or the subroutine (lower level) will generate the same result.

Because the present invention restricts network based communication of selected processes to a specific set of one or more network addresses, the present invention may be readily utilized by ISP's to provide multiple, commercially viable server programs executing on a on a single physical host computer.

It is to be understood that one desirable embodiment of the present invention operates externally to all other processes. The present invention detects when a process is attempting to designate a network address for subsequent communication, and when a process is attempting to conduct network-address based communication that does not require the pre-designation of a network address. If the process attempting the communication is associated with a set of network addresses, the communication is only allowed to proceed via an associated address. Therefore, the present invention can ensure

that unauthorized server programs (processes) and third parties are prevented from monitoring communication of other server programs.

It is also to be understood that the present invention does not require that the source code of the selected processes be rewritten, or that the processes themselves be upgraded in any way. The present invention can restrict the communication of any process, including all existing processes. Therefore, the present invention can be used to restrict the network address-based communication of server programs without requiring that the server programs be rewritten or upgraded.

It will be readily apparent to one skilled in the art that the present invention can be utilized to restrict process communication of any type of process to a specific set of network addresses. Thus, although restricting network address-based communication of server programs is one important function of the present invention, the present invention is by no means so limited. For example, the present invention can also be utilized to restrict network address-based communication of client processes, communication daemons, device drivers, and the like. All such uses are, of course, within the scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a system for restricting process communication to a set of specific network addresses according to one embodiment of the present invention.

Fig. 2A is a high level flowchart illustrating the steps performed in order to restrict communication of selected processes to single network addresses, according to one embodiment of the present invention.

Fig. 2B is a block diagram illustrating a system for restricting communication of selected processes to single network addresses, according to the embodiment of the present invention depicted in FIG. 2A.

Fig. 3A is a high level flowchart illustrating the steps performed in order to restrict communication of selected processes to a set of specific, multiple addresses, according to another embodiment of the present invention.

Fig. 3B is a block diagram illustrating a system for restricting communication of selected processes to a set of specific, multiple addresses, according to the embodiment of the present invention depicted in FIG. 3A.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

I. SYSTEM OVERVIEW

FIG. 1 illustrates a system 100 for restricting process communication to a set of specific network addresses according to one embodiment of the present invention. A

5   computer memory 101 includes user address space 103 and operating system address space 105. A process 107 executes in user address space 103. Although FIG. 1 illustrates only a single process 107 executing in user address space 103, it is to be understood that within a given computer memory 101, multiple processes 107 can execute simultaneously. The computer memory 101 is preferably part of an otherwise

10   conventional computer system, including at least one central processing unit, at least one static storage device, and input/output devices.

An operating system kernel 109 executes in operating system address space 105. Techniques known in the art are utilized to insert an interception module 111 into the operating system 117. In a preferred embodiment, an interception module is dynamically

15   loaded 111 into the operating system kernel 109, while the kernel 109 is active. The interception module 111 is preferably in the form of object code, the functional features of which are described in detail below. Preferably, a single interception module 111 that includes all of the object code described below is loaded into the operating system kernel 109. In alternative embodiments, multiple interception modules 111 are loaded, each

20   module 111 including a subset of the object code.

Also inserted into the operating system 117 is an association table 127, which will be used to store associations 129 between selected processes 107 and sets of network

addresses. Preferably, the association table 127 is loaded into the operating system kernel 109, while the kernel 109 is active. In alternative embodiments, the associations are not loaded into the kernel 109, but are stored in user address space 103 or operating system address space 105 as desired. In alternative embodiments of the present

5   invention, the associations 129 are stored in a format other than a table 127, for example a linked list.

Processes 107 that are selected to have their network address-based communication restricted to a set of specific network addresses are loaded into process address space 119 by a modified loader program 121. As explained above, a loader

10   program is an operating system utility that is used to execute computer programs that are stored on static media. A loader program typically executes in user address space 103. When a user attempts to execute a computer program (for example by typing the name of an executable file at a command line, or by clicking on an icon associated with the program), the loader program executes and proceeds to load an executable image from

15   static media into process address space 119, and then to initiate execution of the loaded image by transferring execution to the first instruction thereof.

The present invention utilizes a modified loader program 121 to load selected processes 107 that are to have their network-address based communication restricted to a set of specific network addresses. Like a standard loader program, the modified loader

20   121 loads executable images from static media into process address space 119. The modified loader program 121 proceeds to store, in the association table (or alternative data structure) an association 129 between the process identifier (or in alternative embodiments, alternative process identifying data such as process name) of the loaded

process 107 and the set of specific network addresses for the process 107. Once the association 129 has been stored, the loader program 121 initiates execution of the process 107 by transferring execution to the first instruction thereof.

As stated above, only selected processes 107 are loaded by the modified loader program 121. Non-selected processes are loaded with the standard, default operating system loader program, which simply loads and executes the process. Thus, non-selected processes are not associated with sets of network addresses.

The loading of selected processes 107 and non-selected processes with two different loader programs is possible because multitasking operating systems such as UNIX ® allow the use of multiple loader programs. The decision as to which processes 107 are to be loaded with the modified loader program 107 can be made by a system administrator, or by a user. A system administrator can limit access to the modified loader program 121, and thus limit the ability of users to specify which processes will be selected.

In an alternative embodiment of the present invention, a single, modified loader program 121 is utilized to load both selected processes 107 and non-selected processes. In that embodiment, a list of selected processes 107 is stored in computer memory 101. The list is preferably dynamic, and can be updated with additions or deletions as desired. The modified loader program 121 utilizes the list to determine if a process to be loaded is a selected process 107. If so, the modified loader program 121 loads the selected process 107, and stores the association 129, as described above. If the process is not selected, the modified loader 121 simply loads the process in the manner of a default loader program.

In order to restrict the communication of selected processes 107 to specific sets of network addresses, the present invention intercepts certain system calls 115. Pointers 114 to system calls 115 are located in an operating system interrupt vector table 113. It is to be understood that the term "interrupt vector table" as used herein denotes an area in operating system address space 105 in which there are stored the addresses of system calls. In the UNIX ® operating system, this part of the operating system is called the "interrupt vector table," and that term is used in this specification. Other operating systems employ different terminology to denote the same system component. An interrupt vector table by any other name is still within the scope of the present invention.

A copy 116 is made of a pointer 114 to each system call 115 to be intercepted. These copies 116 of pointers 114 are preferably stored in operating system address space 105, but in an alternative embodiments are stored in user address space 103. Once the copies 116 have been saved, the pointers 114 in the interrupt vector table 113 to the system calls 115 to be intercepted are replaced with pointers 118 to the interception module 111, such that when a system call 115 to be intercepted is made, the interception module 111 executes instead. In one embodiment of the present invention, this copying, storing, and replacing of pointers is performed by the interception model 111. In other embodiments, copying, storing, and replacing of pointers is performed by a pointer management module executing in either operating system address space 105 or user address space 103 as desired. The pointer management module can either be a stand alone program, or a component of a larger application program as desired.

Executing alternative code when a system call 115 is made comprises intercepting the system call 115. The steps of inserting an interception module 111 into the operating

system 117, making a copy 116 of an operating system pointer 114 to a system call 115, and replacing the operating system pointer 114 with a pointer 118 to the interception module 111 facilitate interception of a system call 115. When a call is made to a system call 115 to be intercepted, the operating system 117 uses the pointer 118 in the interrupt

5   vector table 113 to the interception module 111 to execute the interception module 111.

It is to be understood that the term "interception module" is used herein to denote alternative object code that executes in response to the making of a system call. The alternative object code that executes when a system call 115 is made is sometimes referred to as a "system call wrapper" as opposed to an "interception module." Of

10   course, the execution of alternative object code in response to the making of a system call is within the scope of the present invention regardless of what the alternative object code is called.

It is to be understood that in the present invention, not all system calls 115 need be intercepted. Only pointers 114 to system calls 115 to be intercepted are replaced with

15   pointers 118 to the interception module 111. Pointers 114 to system calls 115 which are not to be intercepted are not replaced. Thus, when a non-intercepted system call 115 is made, the system call 115 executes, not the interception module 111.

Additionally, in order to restrict the communication of selected processes 107 to specific sets of network addresses, a communication protocol stack 131 is modified so as

20   to intercept certain communication subroutines 139. It is to be understood that the term "communication protocol stack" is used herein to denote the layers of software (e.g. transport layer, internetwork layer, hardware interface layer) that facilitate inter-computer and inter-process communication according to a specific protocol. The layers of software

that make up a communication protocol stack 131 are typically stored in operating system address space 105. These layers of software are sometimes referred to by other names, for example "communication protocol suite." Of course, the present invention is not limited to any specific name.

5   A communication protocol stack 131 contains pointers 133 to subroutines 139 that perform various communication tasks. Non-exhaustive examples of such subroutines 139 include a subroutine that establishes a communication channel, a subroutine that binds a communication channel to a network address, and a subroutine that transmits data via a communication channel.

10   The present invention intercepts certain ones of these subroutines 139. To prepare to intercept subroutines 139, a copy 137 of a pointer 133 to each subroutine 139 to be intercepted is made. These copies 137 are preferably stored in operating system address space 105, but in an alternative embodiments are stored in user address space 103. Once the copies 137 have been made and saved, the pointers 133 in the

15   communication protocol stack 131 to the subroutines 139 to be intercepted are replaced with pointers 135 to the interception module 111, such that when a subroutine 139 to be intercepted is called, the interception module 111 executes instead. In one embodiment of the present invention, the modification of the communication protocol stack 131, including the copying, storing, and replacing of pointers, is performed by the interception

20   model 111. In other embodiments, the modification of the communication protocol stack 131 is performed by a communication protocol stack modification module executing in either operating system address space 105 or user address space 103 as desired. The pointer management module can either be a stand alone program, or a component of a

larger application program as desired. In one embodiment, the communication protocol stack modification module is the same software module as the pointer management module.

Executing alternative code when a subroutine 139 is called comprises intercepting the subroutine 139. The steps of inserting an interception module 111 into the operating system 117, making a copy 137 of a communication protocol stack 131 pointer 133 to a subroutine 139, and replacing the communication protocol stack 131 pointer 133 with a pointer 135 to the interception module 111 facilitate interception of a subroutine 139. When a call is made to a subroutine 139 to be intercepted, the operating system 117 uses the pointer 135 in the communication protocol stack 131 to the interception module 111 to execute the interception module 111.

It is to be understood that in the present invention, not all subroutines 139 need be intercepted. Only pointers 133 to subroutines 139 to be intercepted are replaced with pointers 135 to the interception module 111. Pointers 133 to subroutines 139 that are not to be intercepted are not replaced. Thus, when a non-intercepted subroutine 139 is made, the subroutine 139 executes, not the interception module 111.

It is also to be understood that a single operating system 117 generally includes multiple communication protocol stacks 131, each protocol stack 139 facilitating communication according to a specific protocol. In one preferred embodiment of the present invention, the communication protocol stack 139 that is modified is the Transport Control Protocol (TCP/IP) stack 139. Other embodiments modify other protocol stacks 139 of other communication protocols as desired, for example User Data Protocol (UDP) or Internet Control Message Protocol (ICMP).

II. RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO SINGLE NETWORK ADDRESSES

FIG. 2A illustrates the steps performed in order to restrict communication of selected processes 107 to single network addresses, according to one embodiment of the present invention. The modified loader program 121 loads 251 selected processes 107. For each selected process 107, the modified loader 121 stores 253 an association 201 between the process 107 and a specific network address via which the process 107 is permitted to communicate. In order to keep the associations 201 complete and current, system calls 115 that create child processes 107 and system calls 115 that delete processes 107 are intercepted 255, 259. If a selected process 107 creates a child process 107, an association 201 is created 257 between the child process 107 and the network address via which the parent process 107 is permitted to communicate. If a selected process 107 is deleted, the association 201 between the deleted process 107 and a network address is also deleted 261. The stored associations 201 are then utilized to restrict the selected processes 107 to communication via their associated specific network addresses. Certain network address-based communication protocol subroutines 139 are intercepted 263 in order to detect attempts by selected processes 107 to designate a network address for subsequent communication, and to detect attempts to communicate without having designated a specific network address. When such attempts are detected, the associations 201 are examined 265, and the selected processes 107 are restricted 267 to communication via their associated addresses.

FIG 2B illustrates a system 200 for restricting communication of selected processes 107 to single network addresses, according to the embodiment of the present invention illustrated in FIG. 2A. In the embodiment depicted in FIG. 2A and and FIG.

2B, each selected process 107 is restricted to network address-based communication via a single associated address.

a. Loading a Selected Process

Each selected process 107 is loaded into user address space 103 by the modified
5   loader program 121. For each selected process 107, the modified loader stores, in the association table 127, an association 201 between the process and a single, specific network address. The selected process 107 will be restricted to the use of this network address for all network address-based communication. Additionally, the loader program stores an association 202 between the process 107 and a dedicated local host address.
10 The dedicated local host address is associated only with the selected process 107, and only for intra-computer communication. Recall that on a dedicated physical host computer, only one local host address is necessary because only a single server program executes on that computer. All intra-computer communication with the server can be conducted via the single local host address. However, where multiple server programs
15 (processes 107) execute on a single physical computer (or in any application where intra-computer communication can be targeted to a plurality of processes 107), each such process 107 must have its own local host address via which it can conduct intra-computer communication. Thus, in the embodiment depicted in FIG. 2A and Fig. 2B, each selected process 107 is associated with a single, dedicated local host address via which the process
20 107 can conduct intra-computer communication, in addition to being associated with a single, specific network addresses via which the process 107 can conduct inter-computer communication.

b. Intercepting System Calls

In the embodiment depicted in FIG. 2A and Fig. 2B, all system calls 115 that establish a child process are intercepted (for example, the UNIX ® spawn function). If a process 107 is restricted to communication via a single network address, it is necessary to

5   so restrict all child processes 107 created by the process 107. Otherwise, a process 107 could effectively communicate via a non-associated network address by creating a child process 107, and instructing the child process to communicate via the non-associated network address. In order to restrict all progeny of selected processes 107, all system calls 115 that establish a child process are intercepted.

10   When a system call 115 that establishes a child process is made, the operating system 117 uses the pointer 118 in the interrupt vector table 113 to execute the interception module 111. The interception module 111 examines the association table 127 to determine whether the process 107 that is attempting to establish a child process is associated with a network address. If so, the interception module 111 first utilizes the

15   saved copy of the pointer 116 to the system call 303 to make the system call 303. The system call 303 establishes the child process 107 (in an alternative embodiment, the interception module 111 establishes the child process 107 itself, rather than making the system call 115). When the system call 303 terminates, the interception module 111 stores, in the association table 127, an association 201 between the child process 107 and

20   the single, network address with which the parent process 107 is associated. The child process 107 will then be restricted to network address based communication via this associated network address. The interception module 111 also stores, in the association table 127, an association 202 between the child process 107 and the local host address

with which the parent process 107 is associated. The child process 107 will then be restricted to intra-computer communication via the associated local host address. If the process 107 that is attempting to establish a child process 107 is not associated with a network address, the interception module 111 simply makes the system call 115 such that

5   execution control returns to the calling process 107 after the system call 115 executes.

In order to keep the association table 127 current, the present invention also intercepts system calls 115 that delete (terminate) a process 107 (for example, the UNIX ® exit function). Whenever a process 107 makes a system call 115 to delete a process 107, the interception module 111 executes instead. The interception module 111

10 examines the association table 127 to determine whether the process 107 to be deleted is associated with a network address. If so, the interception module 111 utilizes the copy of the pointer 116 to execute the system call 115. The system call 115 deletes the process 107 (in an alternative embodiment, the interception module 111 deletes process 107 itself, rather than making the system call 115). When the system call 115 exits, the

15 interception module 111 continues to execute, and deletes the association 201 in the association table 127 between the deleted process 107 and the network address. The interception module 111 also deletes the association 202 between the deleted process 107 and its dedicated local host address. This is appropriate, because the process 107 no longer exists. If the process 107 to be deleted is not associated with a network address,

20 the interception module 111 simply makes the system call 115 such that execution control returns to the calling process 107 after the system call 115 executes.

c. Maintaining a Complete Association Table

Every selected process 107 is loaded by the modified loader program 121, which stores an association between the process 107 and the single network address via which the process 107 is permitted to communicate. Every system call 115 that establishes a child process 107 is intercepted. If a selected process 107 establishes a child process 107, an association 201 between the child process 107 and the associated network address of the parent process 107 is stored. Additionally, every system call 115 that deletes a process 107 is intercepted. If a deleted process 107 is associated with a network address, the association 201 is deleted. Thus, the association table 127 includes an association 201 between every selected process 107 and the single network address via which the selected process 107 is permitted to communicate. Because the association table 127 includes an association 201 for every selected process, the association table 127 can be utilized to determine whether or not a specific process 107 is selected, and if so to restrict the communication of that process to the associated network address.

d. Detecting Attempted Communication

Network address-based communication of selected processes 107 is restricted to the associated network addresses. In order to restrict selected processes 107 to communication via associated addresses, certain attempts by processes 107 to facilitate network address-based communication, and certain attempts by processes 107 to communicate via a network address are detected. Specifically detected are attempts by processes 107 to designate a network address for subsequent communication, and attempts by processes 107 to communicate without first designating a specific network address. When such attempts are detected, the association table 127 is examined to

determine whether the process 107 is a selected process 107. If so, it is ensured that all communication is via an associated address.

i. Attempts to Designate a Network Address for Communication

In order to detect when a process 107 attempts to designate a network address for subsequent communication, the present invention intercepts certain communication protocol subroutines 139 that facilitate network address-based communication. Specifically, the present invention intercepts subroutines 203 that associate a communication channel with a network address (for example, the TCP/IP bind function). Under TCP/IP and other communication protocols, many communication functions can not be executed until a process 107 has first associated a communication channel with a network address via which the communication is to occur. If a process 107 attempts to utilize such a function to communicate via a network address without first having associated a communication channel with the network address, the function will not execute the communication, but will instead generate an error. By intercepting subroutines 203 that associate a communication channel with a network address, the present invention can ensure that no selected process 107 associates a communication channel with a non-associated network address. This further ensures that no selected process 107 communicates via a non-associated network address by utilizing a communication function that requires that a communication channel be associated with the network address.

It is to be understood that by the term "communication channel" it is meant a logical interface via which communication can be conducted. A communication channel is typically although not necessarily associated with both a network address and a port.

Typically, separate communication channels must be present at each end of a communication session. A communication channel is sometimes denoted by the term "communication channel end" or the term "communication channel end point." Communication channels include but are not limited to TCP/IP sockets. In other words,

5   all TCP/IP sockets are communication channels, but not all communication channels are TCP/IP sockets.

       In order to intercept subroutines 203 that associate a communication channel with a network address, a communication protocol stack 131 is modified so as to intercept these subroutines 203. In one preferred embodiment, the TCP/IP stack is modified. In

10   other embodiments, the protocol stacks of communication protocols are modified as desired. Regardless, the communication protocol stack 131 to be modified contains at least one pointer 133 to at least one subroutine 203 that associates a communication channel with a network address. The present invention intercepts such subroutines 203. When a process 107 calls a subroutine 203 to associate a communication channel with a

15   network address, the operating system 117 uses the pointer 118 in the communication protocol stack 131 to execute the interception module 111. The interception module 111 examines the association table 127 to determine whether the process 107 that is attempting to associate a communication channel with a network address is a selected process (in other words, a determination is made as to whether the process 107 is

20   associated with a specific address). If the process 107 is not associated with a network address, the interception module 111 simply calls the subroutine 203 such that execution control returns to the calling process 107 after the subroutine 203 executes.

If the process 107 is selected, the interception module 111 examines the association table to determine whether the network address that the process 107 is attempting to associate a communication channel with is associated with the process 107. If the address is associated with the process 107, then the process 107 is permitted to

5   communicate via the network address. In this case, the interception module 111 calls the subroutine 203. The subroutine 203 associates the communication channel with the network address to allow future communication via the network address by the process 107 that called the subroutine 203. Execution control then returns to the calling process 107 after the subroutine 203 executes. In an alternative embodiment, rather than calling

10   the subroutine 203 the interception module 111 associates the communication channel with the network address associated with the process 107, and then returns execution control to the calling process 107.

If the network address that the process 107 is attempting to associate a communication channel with is not associated with the process 107, another possibility is

15   that the process 107 is attempting to associate a communication channel with the local host network address. If the address the process 107 is attempting to associate a communication channel with is the local host address of the physical host computer (for example, under TCP/IP 127.0.0.1), the interception module 111 associates the communication with the dedicated local host address of the process 107, and then returns

20   execution control to the calling process 107. The process 107 can proceed to conduct intra-computer communication via its dedicated local host address. If the address the process 107 is attempting to associate a communication channel with the dedicated local host address of the process 107, the interception module 111 calls the subroutine 203.

The subroutine 203 associates the communication channel with the dedicated local host address to allow future intra-computer communication via the dedicated local host address by the process 107 that called the subroutine 203. Execution control then returns to the calling process 107 after the subroutine 203 executes. In an alternative

5   embodiment, rather than calling the subroutine 203 the interception module 111 associates the communication channel with the dedicated local host address of the process 107, and then returns execution control to the calling process 107.

If the network address that the process 107 is attempting to associate a communication channel with is not associated with the process 107 and is not a local host

10   address, yet another possibility is that the process 107 is attempting to associate a communication channel with a wildcard network address. It is possible for a process 107 to call a subroutine 203 that associates a communication channel with a network address, and pass the subroutine 203 a wildcard as the parameter that indicates the network address. Under TCP/IP, a wildcard is indicated by passing the value 0 (typically

15   represented by a constant such as "INADDR_ANY") as a parameter. Passing a wildcard constitutes a request that the communication channel be automatically associated with a network address. Typically, the communication channel is associated with the target network address of the next incoming communication request. This could be any network address associated with the physical host. Because the selected process 107 is

20   restricted to communication via the associated network address, association of the communication channel with any address associated with the physical host is not permitted. Therefore, when the interception module 111 detects that a selected process 107 is attempting to associate a communication channel with a wild card network

address, the interception module 111 associates the communication channel with the network address with which the process is associated. The process 107 is only permitted to conduct inter-computer communication via the single associated network address. Thus, when the process 107 requests that a communication channel be automatically

5   associated with a network address, the interception module 111 associates the communication channel with the network address with which the process 107 is associated. After associating the communication channel with the network address, the interception module 111 returns execution control to the calling process 107.

If the process 107 is attempting to associate a communication channel with any

10   network address other than the network address with which the process is associated, a local host address, or a wildcard address, the interception module 111 generates an error condition. In one embodiment, the interception module 111 generates an error condition by throwing an exception. In another embodiment, the interception module 111 generates an error condition by returning an error code to the process 107 that called the

15   subroutine 203. Regardless, the process 107 is unable to communicated via the unassociated network address.

ii. Attempts to Communicate without Having Designated an Address

In order to detect when a process 107 attempts network address based communication without having first designated a network address, the present invention

20   also intercepts subroutines 205 that facilitate network address-based communication without requiring that a communication channel first be associated with a network address. Certain subroutines allow network addressed-based communication without requiring that the process 107 first associate a communication channel with a network

address. Non-exhaustive examples of such subroutines 205 include the TCP/IP connect

function, sendto function, and sendmessage function. The connect function attempts to

establish a connection between a local communication channel and a remote computer.

The sendto and sendmessage functions attempt to transmit data (send packets) to a

5   remote computer. These are only examples of subroutines 205 that facilitate network

address-based communication without requiring that a communication channel first be

associated with a network address. Of course, the present invention is not limited in

scope to these specific examples.

Generally, subroutines 205 that facilitate network address-based communication

10   without requiring that the process 107 first associate a communication channel with a

network address do require that the calling process 107 specify a communication channel.

If the specified communication channel has already been associated with a network

address, subsequent communication will be conducted via that address. However, if the

communication channel has not already been associated with a network address, the

15   communication channel is automatically associated with a randomly chosen network

address that is available for inter-computer communication. This could be any network

address associated with the physical host. Because the selected process 107 is restricted

to communication via a single, specific network address, association of the

communication channel with a randomly selected network address associated with the

20   physical host is not acceptable. To prevent this unacceptable random association, the

present invention intercepts subroutines 205 that facilitate network address-based

communication without requiring that a communication channel first be associated with a

network address.

The communication protocol stack 131 is modified so as to intercept subroutines 205 that facilitate network address-based communication without requiring that a communication channel first be associated with a network address. When a process 107 calls a subroutine 205 that that facilitates network address-based communication without requiring that a communication channel first be associated with a network address, the operating system 117 uses the pointer 118 in the communication protocol stack 131 to execute the interception module 111. The interception module 111 examines the association table 127 to determine whether the process 107 that called the subroutine 205 is a selected process. If the process 107 is not associated with a network address, the interception module 111 simply calls the subroutine 205 such that execution control returns to the calling process 107 after the subroutine 205 executes.

If the process 107 is selected, the interception module 111 first determines whether or not the communication channel that was passed to the subroutine 205 is already associated with a network address. If the communication channel is not already associated with a network address, the interception module 111 associates the communication channel with the network address via which the process 107 is permitted to communicate. The interception module 111 calls the subroutine 205, which facilitates the network address-based communication by the process 107 (alternatively the interception module 111 facilitates the network address-based communication by the process 107 itself, as opposed to calling the subroutine 205). The communication channel has now been associated with the network address via which the process 107 is permitted to communicate. Therefore, the subsequent network address-based

communication facilitated by the subroutine 205 (or interception module 111) will be via this permitted network address.

If the communication channel is already associated with a network address, the interception module 111 determines whether or not that network address is the network address via which the process 107 is permitted to communicate. If the network address with which the communication channel is associated is the network address via which the process 107 is permitted to communicate, the interception module 111 calls the subroutine 205. The subroutine 205 facilitates the network address-based communication via the network address (alternatively, the interception module 111 facilitates the network address-based communication itself, as opposed to calling the subroutine 205). Execution control then returns to the calling process 107.

If the network address with which the communication channel is associated is not the network address via which the process 107 is permitted to communicate, the interception module 111 generates an error condition. In one embodiment, the interception module 111 generates an error condition by throwing an exception. In another embodiment, the interception module 111 generates an error condition by returning an error code to the process 107 that called the subroutine 205. Regardless, the process 107 is unable to communicated via the unassociated network address.

III. RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO SPECIFIC, MULTIPLE NETWORK ADDRESS

FIG. 3A illustrates the steps performed in order to restrict communication of selected processes to a set of specific, multiple addresses, according to another embodiment of the present invention. The modified loader program 121 loads 251

selected processes 107. For each selected process 107, the modified loader 121 stores 351 an association 301 between the process 107 and a set of network addresses via which the process 107 is permitted to communicate. In order to keep the associations complete and current, system calls 115 that create child processes 107 and system calls 115 that

5   delete processes 107 are intercepted 255, 259. If a selected process 107 creates a child process 107, an association 301 is created 353 between the child process 107 and the set of network addresses via which the parent process 107 is permitted to communicate. If a selected process 107 is deleted, the association 301 between the deleted process 107 and a set of network addresses is also deleted 355. The stored associations 301 are then

10   utilized to restrict the selected processes 107 to communication via their associated sets of specific network addresses. Certain network address-based communication protocol subroutines 139 are intercepted 263 in order to detect attempts by selected processes 107 to designate a network address for subsequent communication, and to detect attempts to communicate without having first designated a specific network address. When such

15   attempts are detected, the associations 301 are examined 265, and the selected processes 107 are restricted 267 to communication via the associated addresses.

a. Loading a Selected Process

Fig. 3B illustrates a system 300 for restricting communication of selected processes to a set of specific, multiple addresses, according to another embodiment of the

20   present invention. In the embodiment depicted in FIG. 3A and FIG. 3B, each selected process 107 is loaded into user address space 103 by the modified loader program 121. For each selected process 107, the modified loader stores, in the association table 127, an association 301 between the process and a set of specific, multiple network address. The

selected process 107 will be restricted to the use of this set of network addresses for all network address-based communication. The loader program also stores an association 202 between the process 107 and a dedicated local host address, as in the embodiment depicted in FIG. 2A and Fig. 2B. Thus, in the embodiment depicted in FIG. 3A and Fig.

5   3B, each selected process 107 is associated with a single, dedicated local host address via which the process 107 can conduct intra-computer communication, in addition to being associated with a set of specific, multiple network addresses via which the process 107 can conduct inter-computer communication.

b. Intercepting System Calls

10   As in the embodiment depicted in FIG. 2A and Fig. 2B, in the embodiment depicted in FIG. 3A and Fig. 3B all system calls 115 that establish a child process are intercepted. When a system call 115 that establishes a child process is made, the operating system 117 uses the pointer 118 in the interrupt vector table 113 to execute the interception module 111. The interception module 111 examines the association table

15   127 to determine whether the process 107 that is attempting to establish a child process is associated with a set of network addresses. If so, the interception module 111 first utilizes the saved copy of the pointer 116 to the system call 303 to make the system call 303. The system call 303 executes, thereby establishing the child process 107.

When the system call 303 terminates, the interception module 111 continues to

20   execute. The interception module 111 stores, in the association table 127, an association 301 between the child process 107 and the set of specific, multiple network address with which the parent process 107 is associated. The child process 107 will then be restricted to network address based communication via this set of associated network addresses.

The interception module 111 also stores, in the association table 127, an association 202 between the child process 107 and the local host address with which the parent process 107 is associated. The child process 107 will then be restricted to intra-computer communication via the associated local host address. If the process 107 that is attempting

5   to establish a child process 107 is not associated with a network address, the interception module 111 simply makes the system call 115 such that execution control returns to the calling process 107 after the system call 115 executes.

As in the embodiment depicted in FIG. 2A and Fig. 2B, in order to keep the association table 127 current, in the embodiment depicted in FIG. 3A and Fig. 3B system

10   calls 115 that delete a process 107 are also intercepted. Whenever a process 107 makes a system call 115 to delete a process 107, the interception module 111 executes instead. The interception module 111 examines the association table 127 to determine whether the process 107 to be deleted is associated with a network address. If so, the interception module 111 utilizes the copy of the pointer 116 to execute the system call 115. The

15   system call 115 executes and deletes the process 107. When the system call 115 exits, the interception module 111 continues to execute. The interception module 111 deletes the association 301 in the association table 127 between the deleted process 107 and the set of network addresses. The interception module 111 also deletes the association 202 between the deleted process 107 and its dedicated local host address. If the process 107

20   to be deleted is not associated with a network address, the interception module 111 simply makes the system call 115 such that execution control returns to the calling process 107 after the system call 115 executes.

c. Maintaining a Complete Association Table

Every selected process 107 is loaded by the modified loader program 121, which stores an association 301 between the process 107 and a set of specific, multiple network addresses via which the process 107 is permitted to communicate. Every system call 115 that establishes a child process 107 is intercepted. If a selected process 107 establishes a child process 107, an association 301 between the child process 107 and the associated set of network addresses of the parent process 107 is stored. Additionally, every system call 115 that deletes a process 107 is intercepted. If a deleted process 107 is associated with a set of network addresses, the association 301 is deleted. Thus, the association table 127 includes an association 301 between every selected process 107 and the set of specific, multiple network addresses via which the selected process 107 is permitted to communicate. Because the association table 127 includes an association 301 for every selected process, the association table 127 can be utilized to determine whether or not a specific process 107 is selected, and if so to restrict the communication of that process to the associated set of network addresses.

d. Detecting Attempted Communication

Network address-based communication of the selected processes is restricted to the associated network addresses. In order to restrict selected processes 107 to communication via associated addresses, certain attempts by processes 107 to facilitate network address-based communication, and certain attempts by processes 107 to communicate via a network address are detected. Specifically detected are attempts by processes 107 to designate a network address for subsequent communication, and attempts by processes 107 to communicate without first designating a specific network

address. When such attempts are detected, the association table 127 is examined to determine whether the process 107 is a selected process 107. If so, it is ensured that all communication is via an associated address.

i. Attempts to Designate a Network Address for Communication

5   In order to detect when a process 107 attempts to designate a network address for subsequent communication, the present invention intercepts certain communication protocol subroutines 139 that facilitate network address-based communication. Specifically, as in the embodiment depicted in FIG. 2A and Fig. 2B, in the embodiment depicted in FIG. 3A and Fig. 3B subroutines 203 that associate a communication channel

10   with a network address are intercepted.

When a process 107 calls a subroutine 203 to associate a communication channel with a network address, the operating system 117 uses the pointer 118 in the communication protocol stack 131 to execute the interception module 111. The interception module 111 examines the association table 127 to determine whether the

15   process 107 that is attempting to associate a communication channel with a network address is a selected process. If the process 107 is not associated with a set of network addresses, the interception module 111 simply calls the subroutine 203 such that execution control returns to the calling process 107 after the subroutine 203 executes.

If the process 107 is selected, the interception module 111 examines the

20   association table 127 to determine whether the network address that the process 107 is attempting to associate a communication channel with is one of the addresses in the set associated with the process 107. If the address is in the set, then the process 107 is permitted to communicate via the network address. In this case, the interception module

111 calls the subroutine 203. The subroutine 203 associates the communication channel with the network address to allow future communication via the network address by the process 107 that called the subroutine 203. Execution control then returns to the calling process 107 after the subroutine 203 executes.

5   If the network address that the process 107 is attempting to associate a communication channel with is not in the set associated with the process 107, another possibility is that the process 107 is attempting to associate a communication channel with the local host network address. The embodiment depicted in FIG. 3A and Fig. 3B processes such an attempt in the manner of the embodiment depicted in FIG. 2A and Fig.

10   2B.

       If the network address that the process 107 is attempting to associate a communication channel with is not associated with the process 107 and is not a local host address, yet another possibility is that the process 107 is attempting to associate a communication channel with a wildcard network address. Recall that passing a wildcard

15   to a subroutine 203 that associates a communication channel with a network address constitutes a request to associate a communication channel with any network address available for network address-based communication. Because the process 107 is permitted to conduct network address-based communication via a set of multiple network addresses, each multiple network address of the set is available for network address-

20   based communication. Therefore, it is desirable to associate a communication channel with each of the network address of the set. Thus, when a process 107 attempts to associate a communication channel with a wild card network address, the interception module 111 first associates the communication channel with one of the network

addresses in the set. Next, the interception module 111 creates a communication channel

(socket) for each remaining network address in the set, and proceeds to associate each

created communication channel with one of the remaining network addresses. The result

is that for each network address in the set, there exists a separate, associated

5   communication channel. The resulting communication channel-network address pairs

303 are then stored, preferably in an auxiliary table 305, but alternatively in other data

structures as desired. The stored communication channel-network address pairs 303 are

then available to the process 107 for subsequent network address-based communication.

Also stored (preferably in the auxiliary table 305) is an indicator 309 that the set of

10   communication channel-network address pairs 303 associated with the previously

existing communication channel that the process attempted to associate with a wild card

network address. The purpose of this indicator 309 is explained below.

        If the process 107 is attempting to associate a communication channel with any

network address other than a network address with which the process 107 is associated, a

15   local host address, or a wildcard address, the interception module 111 generates an error

condition. In one embodiment, the interception module 111 generates an error condition

by throwing an exception. In another embodiment, the interception module 111

generates an error condition by returning an error code to the process 107 that called the

subroutine 203. Regardless, the process 107 is unable to communicated via the

20   unassociated network address.

ii. Attempts to Communicate without Having Designated an Address

        As in the embodiment depicted in FIG. 2A and Fig. 2B, in order to detect when a

process 107 attempts network address based communication without having designated a

network address, subroutines 205 that facilitate network address-based communication without requiring that a communication channel first be associated with a network address are also intercepted. When a process 107 calls a subroutine 205 that facilitates network address-based communication without requiring that a communication channel first be associated with a network address, the operating system 117 uses the pointer 118 in the communication protocol stack 131 to execute the interception module 111. The interception module 111 examines the association table 127 to determine whether the process 107 that called the subroutine is a selected process. If the process 107 is not associated with a set of network addresses, the interception module 111 simply calls the subroutine 205 such that execution control returns to the calling process 107 after the subroutine 205 executes.

If the process 107 is selected, the interception module 111 first determines whether or not the communication channel that was passed to the subroutine 205 is already associated with a network address. If the communication channel is not already associated with a network address, the interception module 111 associates the communication channel with a random one of the network addresses via which the process 107 is permitted to communicate. The interception module 111 calls the subroutine 205, which facilitates the network address-based communication by the process 107. The communication channel has now been associated with one of the network addresses via which the process 107 is permitted to communicate. Therefore, the subsequent network address-based communication facilitated by the subroutine 107 will be via this permitted network address.

If the communication channel is already associated with a network address, the interception module 111 determines whether or not that network address is one of the network addresses via which the process 107 is permitted to communicate. If the network address is one of the network addresses via which the process 107 is permitted to communicate, then the interception module 111 calls the subroutine 205. The subroutine 205 facilitates the network address-based communication via the network address. Execution control then returns to the calling process 107 after the subroutine 205 executes.

If the network address with which the communication channel is associated is not one of the network addresses via which the process 107 is permitted to communicate, the interception module 111 generates an error condition. In one embodiment, the interception module 111 generates an error condition by throwing an exception. In another embodiment, the interception module 111 generates an error condition by returning an error code to the process 107 that called the subroutine 205. Regardless, the process 107 is unable to communicated via the unassociated network address.

e. Managing Communication via Multiple Addresses

Recall that a communication channel can be associated with a wildcard network address. An attempt to create such an association is a request by a process 107 that subsequent communication utilizing the communication channel be conducted via any available network address. Such a communication channel can be thought of as a wildcard communication channel. In the embodiment depicted in FIG. 3A and Fig. 3B, when a process 107 attempts to associate a communication channel with a wildcard network address, the interception module 111 associates the communication channel with

a set of communication channel-network address pairs that includes each network address

of the permitted set, as described above. Subsequent communication utilizing the

channel can be via any one of the channel-address pairs that is available. Therefore, in

the embodiment depicted in FIG. 3A and Fig. 3B, certain communication protocol

5   subroutines 139 and certain system calls 115 are intercepted to ensure that

communication utilizing a wildcard communication channel (listening socket) is via any

available address in the set, but not via an address not in the set.

i. Receiving Incoming Requests to Initiate Communication

In the embodiment depicted in FIG. 3A and Fig. 3B, subroutines 307 that receive,

10   on an existing communication channel, an incoming request to initiate a communication

session are intercepted (e.g., the TCP/IP accept function). Under TCP/IP and other

communication protocols, a process 107 can pass such a subroutine 307 either a

communication channel that is associated with a specific network address, or a wildcard

communication channel. When a process 107 passes a wildcard communication channel,

15   the process 107 is requesting that an incoming request be received via whatever network

address is available. In the case of the embodiment depicted in FIG. 3A and Fig. 3B, the

request can be received via any one of the addresses in the set, so it is desirable to

determine which address in the set is available first, and then to receive the request via

that address.

20   When a process 107 calls a subroutine 307 that receives an incoming request to

initiate a communication session, the operating system 117 uses the pointer 118 in the

communication protocol stack 131 to execute the interception module 111. The

interception module 111 determines whether the communication channel that the process

107 passed to the subroutine 307 is a wildcard communication channel. To so determine, the interception module 111 examines the indicator 309 in the auxiliary table 305 to determine whether the communication channel is associated with a set of communication channel-network address pairs 303. If so the channel is a wildcard channel, and an

5   incoming communication request can be received via any one of the communication channel-network address pairs 303. Therefore, the interception module 111 retrieves the pairs 303 from the auxiliary table 305, and proceeds to identify one of the channel-network pairs 303 that is presently ready to receive an incoming communication request. If none of the pairs 303 are ready, the interception module 111 waits until a first one is

10 ready. In order to identify one of the communication channel-network address pairs 303 as being ready to accept an incoming communication request, the interception module 111 preferably calls a communication protocol subroutine 139 that determines which of a specified set of communication channels is ready to receive an incoming communication request (e.g. the TCP/IP select function). Once a pair 303 is identified as ready, the

15 interception module 111 calls the subroutine 307 that receives an incoming request to initiate a communication session, and passes it the communication channel that has been identified as being ready to receive the incoming request. Because the communication channel is ready to receive the incoming request, when the subroutine 307 executes the channel receives the request immediately, and the subroutine 307 proceeds to return

20 control to the calling process 107.

   If the existing communication channel is not a wildcard channel, then the interception module 111 simply calls the subroutine 307 for the process 107, passing it the communication channel.

ii. Identifying Available Communication Channels

Also intercepted are subroutines 311 that identify an available communication channel (e.g., the TCP/IP select function). Under TCP/IP and other communication protocols, a process 107 can pass such a subroutine 311 one or more communication channels, and the subroutine 311 will indicate a first one of the channels to become available for communication. Recall that a wildcard communication channel is associated with a set of communication channel-network address pairs 303. If a process passes a wildcard communication channel to a subroutine 311 that identifies an available communication channel, it is desirable to pass the subroutine 311 not the wildcard communication channel, but instead all of the communication channels of the set of pairs 303 associated with the wildcard channel.

When a process 107 calls a subroutine 311 that identifies an available communication channel, the operating system 117 uses the pointer 118 in the communication protocol stack 131 to execute the interception module 111. The interception module 111 determines whether the process 107 passed a wildcard communication channel to the subroutine 307. If so, the interception module 111 retrieves the set of communication channel-network address pairs 303 associated with the wildcard communication channel. The interception module 111 proceeds to call the subroutine 311, passing it the communication channels of the pairs 303 in place of the wildcard communication channel. In other words, the wildcard communication channel is expanded into all of the communication channels associated therewith.

If the subroutine 311 was passed only a wildcard communication channel, the interception module 111 passes the subroutine 311 all of the associated channels, and the

subroutine 311 indicates a first one to be ready for communication. If the subroutine 311 was passed a wildcard communication channel and other, non-wildcard communication channels, the interception module 311 passes the subroutine 311 all of the channels associated with the wildcard channel, plus the other, non-wildcard channels passed to the subroutine 311. The subroutine 311 then indicates the first one of all of these channels to be ready for communication. If no communication channel passed to the subroutine 311 is a wildcard channel, then the interception module 111 simply calls the subroutine 307 for the process 107.

iii. Duplicating Communication Channels

Also intercepted are system calls 115 that duplicate a communication channel (*e.g.* the UNIX ® dup and dup2 functions). Whenever such a system call 115 is made, the interception module 111 determines whether the channel to be duplicated is a wildcard channel. To so determine, the interception module 111 examines the indicators 309 in the auxiliary table 305 to determine whether or not the channel to be duplicated is associated with a set of channel-address pairs 303. If so, after allowing the system call 115 to execute and duplicate the channel, the interception module 111 stores an indicator 309 in the auxiliary table 305 that the duplicate of the channel is associated with the set of channel-address pairs 303 with which the duplicated wildcard channel is associated. Then, if a process 107 subsequently uses the duplicate of the channel for communication, the interception module 111 will be able to determine that the duplicate is a wild card channel.

If the channel to be duplicated is not a wildcard channel, the interception module 111 simply makes the system call for the process 107.

iv. Closing Communication Channels

Also intercepted are system calls 115 that close a communication channel (*e.g.* the UNIX ® close function). Whenever such a system call 115 is made, the interception module 111 determines whether the communication channel to be closed is a wildcard

5   channel. If so, the interception module 111 determines if any duplicates of the channel have been made. To so determine, the interception module 111 examines the indicators 309 in the auxiliary table 305 to determine if any other channels are associated with the set of channel-address pairs 303 with which the wildcard channel to be closed is associated. If at least one duplicate of the channel exists, the interception module 111

10   takes no action concerning the set of pairs 303. If no duplicates exist, the channel to be closed is the only channel associated with the set of pairs 303, so the interception module 111 deletes the set of pairs 303 from the auxiliary table 305. Regardless, the interception module proceeds to make the system call 115 to close the communication channel for the process 107.

IV. Alternative Embodiments

15

It is to be understood that in one embodiment of the present invention, rather than modifying a communications protocol stack 131, the present invention intercepts certain system calls 115 associated with network address-based communication. Each communication protocol subroutine 139 is associated with a system call 115. A

20   communication protocol stack 131 and its subroutines 139 are at a lower than system calls 115. In order to request a communication task, a process 107 can make a system call 115, which, in turn, calls the subroutine 139 in the appropriate communication

protocol stack 131. Thus, interception of the appropriate system calls 115 achieves the

same result as the modification of the communication protocol stack 131. Thus,

intercepting either the system calls 115 (higher level) or the communication protocol

subroutines 139 (lower level) will generate the same result.

5

What is claimed is:

1   1. A method in a computer system for restricting network address-based

2   communication by selected processes to a set of specific network addresses, the method

3   comprising:

4   associating at least one selected process with at least one network address;

5   determining whether an attempted network address-based communication

6   of a selected process is via an associated address; and

7   in response to a determination that the communication is via an associated

8   address, allowing the communication to proceed.

1   2. The method of claim 1 further comprising:

2   loading at least one selected process into computer memory; and

3   storing at least one association, between the process and at least one

4   network address.

1   3. The method of claim 1 wherein:

2   associations between selected processes and network addresses are stored

3   in an association table in a computer memory of the computer

4   system.

1   4. The method of claim 3 wherein:

2   the association table is stored in operating system address space.

5. The method of claim 1 wherein:

a network address-based communication comprises an attempt to

designate a network address to be used for subsequent

communication.

6. The method of claim 1 wherein:

a network address-based communication comprises an attempt to associate

a communication channel with a network address.

7. The method of claim 1 wherein:

a network address-based communication comprises an attempt to

communicate without designating a network address to be used for

communication.

8. The method of claim 1 wherein:

a network address-based communication comprises an attempt to establish

a connection to a second process.

9. The method of claim 1 wherein:

a network address-based communication comprises an attempt to transmit

data to a second process.

10. The method of claim 9 wherein:

the second process is executing in a computer memory of the computer

system.

1   11. The method of claim 9 wherein:

2   the second process is executing in a computer memory of a second

3   computer system.

1   12. The method of claim 1 further comprising:

2   determining whether an attempted network address-based communication

3   is via an associated address by intercepting system calls that

4   pertain to network address-based communication.

1   13. The method of claim 12 further comprising:

2   storing object code that determines whether an attempted network address-

3   based communication is via an associated network address; and

4   wherein intercepting comprises replacing a pointer to a system call with a

5   pointer to the object code, such that calling the system call causes

6   the object code to execute.

1   14. The method of claim 13 further comprising:

2   loading an interception module into computer memory, the interception

3   module comprising the object code.

1   15. The method of claim 14 wherein:

2   the interception module is loaded into a running operating system kernel.

1   16. The method of claim 13 wherein determining whether an attempted network

2 address-based communication is via an associated network address comprises:

3   examining at least one stored association to determine whether the

4   processes that called the system call is associated with at least one

5   network address; and

6   in response to a determination that the processes is associated with at least

7   one network address, determining whether the attempted

8   communication is via an associated network address.

1   17. The method of claim 1 further comprising:

2   determining whether an attempted network address-based communication

3   is via an associated address by modifying a communication

4   protocol stack so as to intercept communication protocol

5   subroutines that pertain to network address-based communication.

1   18. The method of claim 17 further comprising:

2   storing object code that determines whether an attempted network address-

3   based communication is via an associated network address; and

4   wherein intercepting comprises replacing a pointer to a subroutine with a

5   pointer to the object code, such that calling the subroutine call

6   causes the object code to execute.

1   19. The method of claim 18 further comprising:

2   loading an interception module into computer memory, the interception

3   module comprising the object code.

1   20. The method of claim 19 wherein:

2   the interception module is loaded into a running operating system kernel.

1   21. The method of claim 18 wherein determining whether an attempted network

2 address-based communication is via an associated network address comprises:

3   examining at least one stored association to determine whether the process

4   that called the subroutine is associated with at least one network

5   address; and

6   in response to a determination that the processes is associated with at least

7   one network address, determining whether the attempted

8   communication is via an associated network address.

1   22. The method of claim 17 wherein:

2   the communication protocol stack that is modified is a Transmission

3   Control Protocol/Internet Protocol stack.

1   23. The method of claim 1 further comprising:

2   detecting creation of a child process by a selected process;

3   associating the child process with all network addresses with which the

4   selected process is associated.

1   24. The method of claim 23 further comprising:

2   detecting creation of a child process by intercepting system calls that

3   create child processes.

1   25. The method of claim 24 further comprising:

2   storing object code that detects creation of a child process by a selected

3   process, and that associates the child process with all network

4   addresses with which the selected process is associated; and

5   wherein intercepting comprises replacing a pointer to a system call with a

6   pointer to the object code, such that calling the system call causes

7   the object code to execute.

1   26. The method of claim 25 further comprising:

2   loading an interception module into computer memory, the interception

3   module comprising the object code.

1   27. The method of claim 26 wherein:

2   the interception module is loaded into a running operating system kernel.

1   28. The method of claim 25 wherein associating comprises:

2   storing an association between the child processes and a network address.

1   29. The method of claim 1 further comprising:

2   associating a child process of a selected process with a single network

3   address with which the selected process is associated;

4   determining whether network address-based communication of the child

5   process is via the associated address; and

6   in response to a determination that the communication is via the associated

7   address, allowing the communication to proceed.

1   30. The method of claim 1 further comprising:

2   associating a child process of a selected process with at least two network

3   addresses with which the selected process is associated;

4   determining whether network address-based communication of the child

5   process is via an associated address; and

6   in response to a determination that the communication is via an associated

7   address, allowing the communication to proceed.

1   31. The method of claim 1 further comprising:

2   detecting termination of a selected process; and

3   deleting all associations between the process and network addresses.

1   32. The method of claim 31 further comprising:

2   detecting termination of a selected process by intercepting system calls

3   that terminate processes.

1   33. The method of claim 32 further comprising:

2   storing object code that deletes all associations between a selected process

3   and network addresses; and

4   wherein intercepting comprises replacing a pointer to a system call with a

5   pointer to the object code, such that calling the system call causes

6   the object code to execute.

1   34. The method of claim 33 further comprising:

2   loading an interception module into computer memory, the interception

3   module comprising the object code.

1   35. The method of claim 34 wherein:

2   the interception module is loaded into a running operating system kernel.

1   36. The method of claim 31 wherein deleting comprises:

2   deleting all associations between a selected process and network

3   addresses.

1   37. The method of claim 1 further comprising:

2   in response to a determination that the attempted communication is not via

3   an associated network address, generating an error condition.

1   38. The method of claim 37 wherein:

2   generating an error condition comprises returning an error code.

1   39. The method of claim 37 wherein:

2   generating an error condition comprises throwing an exception.

1   40. The method of claim 37 further comprising:

2   in response to generating an error condition, not allowing the

3   communication to proceed.

1   41. The method of claim 1 wherein the set consists of one network address.

1   42. The method of claim 1 wherein the set consists of at least two network

2   addresses.

1   43. A method in a computer system for restricting network address-based

2   communication by selected processes to a set of specific network addresses, the method

3   comprising:

4   associating at least one selected process with at least one network address;

5   determining whether an attempted network address-based communication

6   of a selected process is via an associated address; and

7   in response to a determination that the attempted communication is not via

8   an associated address, not allowing the attempted communication

9   to proceed.

1   44. A method in a computer system for restricting network address-based

2   communication by selected processes to specific network addresses, the method

3   comprising:

4   associating at least one selected process with at least one network address;

5   detecting an attempt by a selected processes to associate a communication

6   channel with a network address; and

7   determining whether the network address with which the selected process

8   is attempting to associate a communication channel is associated

9   with the selected process.

1   45. The method of claim 44 further comprising:

2   in response to a determination that the network address is associated with

3   the selected process, allowing the communication channel to be

4   associated with the network address.

1   46. The method of claim 44 further comprising:

2   in response to a determination that the network address is not associated

3   with the selected process, not allowing the communication channel

4   to be associated with the network address.

1   47. A method in a computer system for restricting network address-based

2 communication by selected processes to specific network addresses, the method

3 comprising:

4   associating at least one selected process with at least one network address;

5   detecting an attempt by a selected processes to associate a communication

6   channel with a network address, wherein a provided value for the

7   network address comprises a wild card; and

8   associating the communication channel with a network address that is

9   associated with the process.

1   48. The method of claim 47 wherein:

2   the selected process is associated with a single network address; and

3   associating the communication channel with the single network address.

1   49. The method of claim 47 wherein the selected process is associated with

2 multiple network addresses; the method further comprising:

3   associating the communication channel with one of the multiple network

4   addresses, resulting in a communication channel-network address

5   pair;

6   establishing one communication channel per each additional one of the

7   multiple network addresses;

8   associating each established communication channel with one of the

9   multiple network addresses, resulting in additional communication

10   channel-network address pairs; and

11   associating the communication channel with the communication channel,

12   network address pairs.

1   50. A method in a computer system for restricting network address-based

2 communication by selected processes to specific network addresses, the method

3 comprising:

4   associating at least one selected process with a unique local host address;

5   detecting an attempt by a selected process to communicate with a local

6   host; and

7   designating the unique local host address associated with the selected

8   process to be used by the selected process to communicate with the

9   local host.

1   51. A method in a computer system for restricting network address-based

2 communication by selected processes to specific network addresses, the method

3 comprising:

4   associating at least one selected process with at least one network address;

5   detecting an attempt by a selected process to communicate with a second

6   process via a communication channel;

7   determining if the communication channel is associated with a network

8   address; and

9   in response to determining that the communication channel is not

10   associated with a network address, associating the communication

11   channel with a network address that is associated with the process.

1   52. The method of claim 51 further comprising:

2   in response to a determination that the communication channel is

3   associated with a network address that is associated with the

4   selected process, allowing subsequent communication via the

5   communication channel.

1   53. The method of claim 51 further comprising:

2   in response to a determination that the communication channel is

3   associated with a network address that is not associated with the

4   selected process, not allowing subsequent communication via the

5   communication channel.

1   54. A method in a computer system for restricting network address-based

2 communication by selected processes to specific network addresses, the method

3 comprising:

4   associating at least one selected process with at least one network address;

5   detecting an attempt by a selected process to establish a connection

6   between a communication channel and a second process;

7   determining if the communication channel is associated with a network

8   address; and

9   in response to determining that the communication channel is not

10   associated with a network address, associating the communication

11   channel with a network address that is associated with the selected

12   process.

1   55. The method of claim 54 further comprising:

2   in response to a determination that the communication channel is

3   associated with a network address that is associated with the

4   selected process, allowing the connection to be established.

1   56. The method of claim 54 further comprising:

2   in response to a determination that the communication channel is

3   associated with a network address that is not associated with the

4   selected process, not allowing the connection to be established.

1   57. A method in a computer system for efficiently managing communication via

2   a set of specific, multiple network addresses, the method comprising:

3   associating at least one selected process with a set of specific, multiple

4   network addresses;

5   associating a separate communication channel with each one of the

6   multiple network addresses;

7   detecting an attempt by a selected processes to receive an incoming

8   request to initiate a communication session on one of the

9   communication channels;

10   identifying a first one of the communication channels that is ready to

11   receive the incoming request; and

12   allowing reception of the incoming request on the identified

13   communication channel.

1   58. A computer program product for restricting network address-based

2 communication by selected processes to a set of specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for determining whether an attempted network address-

7   based communication of a selected process is via an associated

8   address;

9   program code for, in response to a determination that the communication

10   is via an associated address, allowing the communication to

11   proceed; and

12   a computer readable medium on which the program codes are stored.

1   59. The computer program product of claim 58 further comprising:

2   program code for loading at least one selected process into computer

3   memory; and

4   program code for storing at least one association between the process and

5   at least one network address.

1   60. The computer program product of claim 58 further comprising:

2   program code for determining whether an attempted network address-

3   based communication is via an associated address by intercepting

4   system calls that pertain to network address-based communication.

1   61. The computer program product of claim 58 further comprising:

2   program code for determining whether an attempted network address-

3   based communication is via an associated address by modifying a

4   communication protocol stack so as to intercept communication

5   protocol subroutines that pertain to network address-based

6   communication.

1   62. The computer program product of claim 61 further comprising:

2   program code for storing object code that determines whether an

3   attempted network address-based communication is via an

4   associated network address; and

5   program code for replacing a pointer to a subroutine with a pointer to the

6   object code, such that calling the subroutine call causes the object

7   code to execute.

1   63. The computer program product of claim 62 further comprising:

2   program code for loading an interception module into computer memory,

3   the interception module comprising the object code.

1   64. The computer program product of claim 62 further comprising:

2   program code for examining at least one stored association to determine

3   whether the processes that called the subroutine is associated with

4   at least one network address; and

5   program code for, in response to a determination that the processes is

6   associated with at least one network address, determining whether

7   the attempted communication is via an associated network address.

1   65. The computer program product of claim 58 further comprising:

2   program code for detecting creation of a child process by a selected

3   process; and

4   program code for associating the child process with all network addresses

5   with which the selected process is associated.

1   66. The computer program product of claim 65 further comprising:

2   program code for detecting creation of a child process by intercepting

3   system calls that create child processes.

1   67. The computer program product of claim 66 further comprising:

2   program code for storing object code that detects creation of a child

3   process by a selected process, and that associates the child process

4 with all network addresses with which the selected process is

5 associated; and

6 program code for replacing a pointer to a system call with a pointer to the

7 object code, such that calling the system call causes the object code

8 to execute.

1 68. The computer program product of claim 67 further comprising:

2 program code for loading an interception module into computer memory,

3 the interception module comprising the object code.

1 69. The computer program product of claim 67 further comprising:

2 program code for storing at least one association between the child

3 processes and a network address.

1 70. The computer program product of claim 58 further comprising:

2 program code for detecting termination of a selected process; and

3 deleting all associations between the process and network addresses.

1 71. The computer program product of claim 70 further comprising:

2 program code for detecting termination of a selected process by

3 intercepting system calls that terminate processes.

1 72. The computer program product of claim 71 further comprising:

2 program code for storing object code that deletes all associations between

3 a selected process and network addresses; and

4   program code for replacing a pointer to a system call with a pointer to the

5   object code, such that calling the system call causes the object code

6   to execute.

1   73. The computer program product of claim 72 further comprising:

2   program code for loading an interception module into computer memory,

3   the interception module comprising the object code.

1   74. The computer program product of claim 71 further comprising:

2   program code for deleting all associations between a selected process and

3   network addresses.

1   75. The computer program product of claim 58 further comprising:

2   program code for, in response to a determination that the attempted

3   communication is not via an associated network address,

4   generating an error condition.

1   76. The computer program product of claim 75 further comprising:

2   program code for, in response to generating an error condition, not

3   allowing the communication to proceed.

1   77. A computer program product for restricting network address-based

2 communication by selected processes to a set of specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for determining whether an attempted network address-

7   based communication of a selected process is via an associated

8   address;

9   program code for, in response to a determination that the communication

10   is not via an associated address, not allowing the attempted

11   communication to proceed; and

12   a computer readable medium on which the program codes are stored.

1   78. A computer program product for restricting network address-based

2 communication by selected processes to specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for detecting an attempt by a selected processes to associate

7   a communication channel with a network address;

8   program code for determining whether the network address with which the

9   selected process is attempting to associate a communication

10   channel is associated with the selected process; and

11   a computer readable medium on which the program codes are stored.

1   79. The computer program product of claim 78 further comprising:

2   program code for, in response to a determination that the network address

3   is associated with the selected process, allowing the

4   communication channel to be associated with the network address.

1   80. The computer program product of claim 78 further comprising:

2   program code for, in response to a determination that the network address

3   is not associated with the selected process, not allowing the

4   communication channel to be associated with the network address.

1   81. A computer program product for restricting network address-based

2 communication by selected processes to specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for detecting an attempt by a selected processes to associate

7   a communication channel with a network address, wherein a

8   provided value for the network address comprises a wild card;

9   program code for associating the communication channel with a network

10   address that is associated with the process; and

11   a computer readable medium on which the program codes are stored.

1   82. The computer program product of claim 81 further comprising:

2   program code for associating the communication channel with a single

3   network address with which the selected process is associated.

1 83. The computer program product of claim 81 wherein the selected process is

2 associated with multiple network addresses; the computer program product further

3 comprising:

4   program code for associating the communication channel with one of the

5   multiple network addresses, resulting in a communication channel-

6   network address pair;

7   program code for establishing one communication channel per each

8   additional one of the multiple network addresses;

9   program code for associating each established communication channel

10   with one of the multiple network addresses, resulting in additional

11   communication channel-network address pairs; and

12   program code for associating the communication channel with the

13   communication channel, network address pairs.

1 84. A computer program product for restricting network address-based

2 communication by selected processes to specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with a unique

5   local host address;

6   program code for detecting an attempt by a selected process to

7   communicate with a local host;

8   program code for designating the unique local host address associated with

9   the selected process to be used by the selected process to

10   communicate with the local host; and

11   a computer readable medium on which the program codes are stored.

1   85. A computer program product for restricting network address-based

2 communication by selected processes to specific network addresses, the computer

3 program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for detecting an attempt by a selected processes to

7   communicate with a second process via a communication channel;

8   program code for determining if the communication channel is associated

9   with a network address;

10   program code for, in response to determining that the communication

11   channel is not associated with a network address, associating the

12   communication channel with a network address that is associated

13   with the process; and

14   a computer readable medium on which the program codes are stored.

1   86. The computer program product of claim 85 further comprising:

2   program code for, in response to a determination that the communication

3   channel is associated with a network address that is associated with

4   the selected process, allowing subsequent communication via the

5   communication channel.

1   87. The computer program product of claim 85 further comprising:

2   program code for, in response to a determination that the communication

3   channel is associated with a network address that is not associated

4   with the selected process, not allowing subsequent communication

5   via the communication channel.

1   88. A computer program product for restricting network address-based

2   communication by selected processes to specific network addresses, the computer

3   program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for detecting an attempt by a selected processes to establish

7   a connection between a communication channel and a second

8   process;

9   program code for determining if the communication channel is associated

10   with a network address;

11   program code for, in response to determining that the communication

12   channel is not associated with a network address, associating the

13   communication channel with a network address that is associated

14   with the selected process; and

15   a computer readable medium on which the program codes are stored.

1	89. The computer program product of claim 88 further comprising:

2	program code for, in response to a determination that the communication

3	channel is associated with a network address that is associated with

4	the selected process, allowing the connection to be established.

1	90. The computer program product of claim 88 further comprising:

2	program code for, in response to a determination that the communication

3	channel is associated with a network address that is not associated

4	with the selected process, not allowing the connection to be

5	established.

1	91. A computer program product for efficiently managing communication via a

2	set of specific, multiple network addresses, the computer program product comprising:

3	program code for associating at least one selected process with a set of

4	specific, multiple network addresses;

5	program code for associating a separate communication channel with each

6	one of the multiple network addresses;

7	program code for detecting an attempt by a selected processes to receive

8	an incoming request to initiate a communication session on one of

9	the communication channels;

10	program code for identifying a first one of the communication channels

11	that is ready to receive the incoming request;

12	program code for allowing reception of the incoming request on the

13	identified communication channel; and

14   a computer readable medium on which the program codes are stored.

1   92. A method in a computer system for restricting network address-based

2   communication by selected processes to a set of specific network addresses, the method

3   comprising:

4   associating at least one selected process with at least one network address;

5   detecting when a selected process attempts to communicate via an

6   unassociated address;

7   not allowing the attempted communication to proceed.

1   93. A computer program product for restricting network address-based

2   communication by selected processes to a set of specific network addresses, the computer

3   program product comprising:

4   program code for associating at least one selected process with at least one

5   network address;

6   program code for detecting when a selected process attempts to

7   communicate via an unassociated address;

8   program code for not allowing the attempted communication to proceed;

9   and

10   a computer readable medium on which the program codes are stored.

1   94. A method in a computer system for efficiently managing communication via

2   a set of specific, multiple network addresses, the method comprising:

3   associating at least one selected process with a set of specific, multiple

4   network addresses;

5   associating a separate communication channel with each one of the

6   multiple network addresses;

7   identifying a first one of the communication channels that is available for

8   communication; and

9   allowing communication to proceed via the communication channel.

1   95. A computer program product for efficiently managing communication via a

2 set of specific, multiple network addresses, the computer program product comprising:

3   program code for associating at least one selected process with a set of

4   specific, multiple network addresses;

5   program code for associating a separate communication channel with each

6   one of the multiple network addresses;

7   program code for identifying a first one of the communication channels

8   that is available for communication;

9   program code for allowing communication to proceed via the

10   communication channel; and

11   a computer readable medium on which the program codes are stored.

Restricting Communication of Selected Processes to a Set of Specific Network Addresses

Abstract of the Disclosure

5   Selected processes are associated with sets of specific network addresses, and the associations are stored. When a selected process creates a child process, an association between the child process and the set of network addresses with which the parent process is associated is stored. When a selected process is deleted, the association between the selected process and its set of network addresses is deleted. Each selected process is

10   restricted to network address-based communication via its associated set of network addresses. Certain communication protocol subroutines associated with network address-based communication are intercepted by an interception module. The interception module detects attempts by selected processes to communicate via network addresses. If a selected process attempts to communicate via an unassociated network addresses, the

15   attempted communication is prohibited.

COMPUTER MEMORY 101

USER ADDRESS SPACE 103

Selected Process 107

Modified Loader Program 121

OPERATING SYSTEM ADDRESS SPACE 105

OPERATING SYSTEM 117

Kernel 109

Interception Module 111

Association 129

Association Table 127

Communication Protocol Stack 131

Pointers to Interception Module 135

Pointers to Subroutines 133

Saved Copies of Pointers 137

Subroutines 139

System Call Vector Table 113

Pointers to Interception Module 118

Pointers to System Calls 114

System Calls 115

Saved Copies of Pointers 116

100

FIG. 1

FIG. 2A

Load Selected Process — 251

Store Association between Process and Specific Address — 253

Intercept System Calls that Create Child Processes — 255

Create Association between Child Process and Specific Address Associated with Parent — 257

Intercept System Calls that Delete Processes — 259

Delete Association between Deleted Process and Associated Network Address — 261

Intercept Communication Protocol Subroutines — 263

Examine Associations of Calling Process — 265

Restrict Network Address-based Communication of Process to Associated Address — 267

FIG. 2A

COMPUTER MEMORY 101

USER ADDRESS SPACE — 103

Selected Process — 107

Modified Loader Program — 121

OPERATING SYSTEM ADDRESS SPACE — 105

OPERATING SYSTEM 117

Kernel — 109

Interception Module — 111

Association between a Process and a Local Host — 202

Association Table — 127

Association between a Process and a Single Network Address — 201

Communication Protocol Stack — 131

Pointers to Interception Module — 135

Pointers to Subroutines — 133

Saved Copies of Pointers — 137

Subroutines — 139

Subroutines that Associate a Communication Channel with a Network Address — 203

Subroutines that Facilitate Network Address-based Communication — 205

System Call Vector Table — 113

Pointers to Interception Module — 118

Pointers to System Calls — 114

System Calls — 115

Saved Copies of Pointers — 116

200

FIG. 2B

```
                  ┌─────────────────────────────┐
                  │     Load Selected Process    │────── 251
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │   Store Association between  │
                  │      Process and Multiple    │────── 351
                  │          Addresses           │
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │     Intercept System Calls   │
                  │   that Create Child Processes │────── 255
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │   Create Association between  │
                  │   Child Process and Multiple  │
                  │  Addresses Associated with    │────── 353
                  │            Parent             │
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │     Intercept System Calls   │
                  │     that Delete Processes     │────── 259
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │   Delete Association between  │
                  │      Deleted Process and      │────── 355
                  │   Associated Network Addresses │
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │    Intercept Communication   │
                  │      Protocol Subroutines    │────── 263
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │    Examine Associations of   │
                  │        Calling Process       │────── 265
                  └─────────────────────────────┘
                                │
                                ▼
                  ┌─────────────────────────────┐
                  │  Restrict Network Address-based │
                  │   Communication of Process to  │────── 267
                  │       Associated Addresses     │
                  └─────────────────────────────┘
```
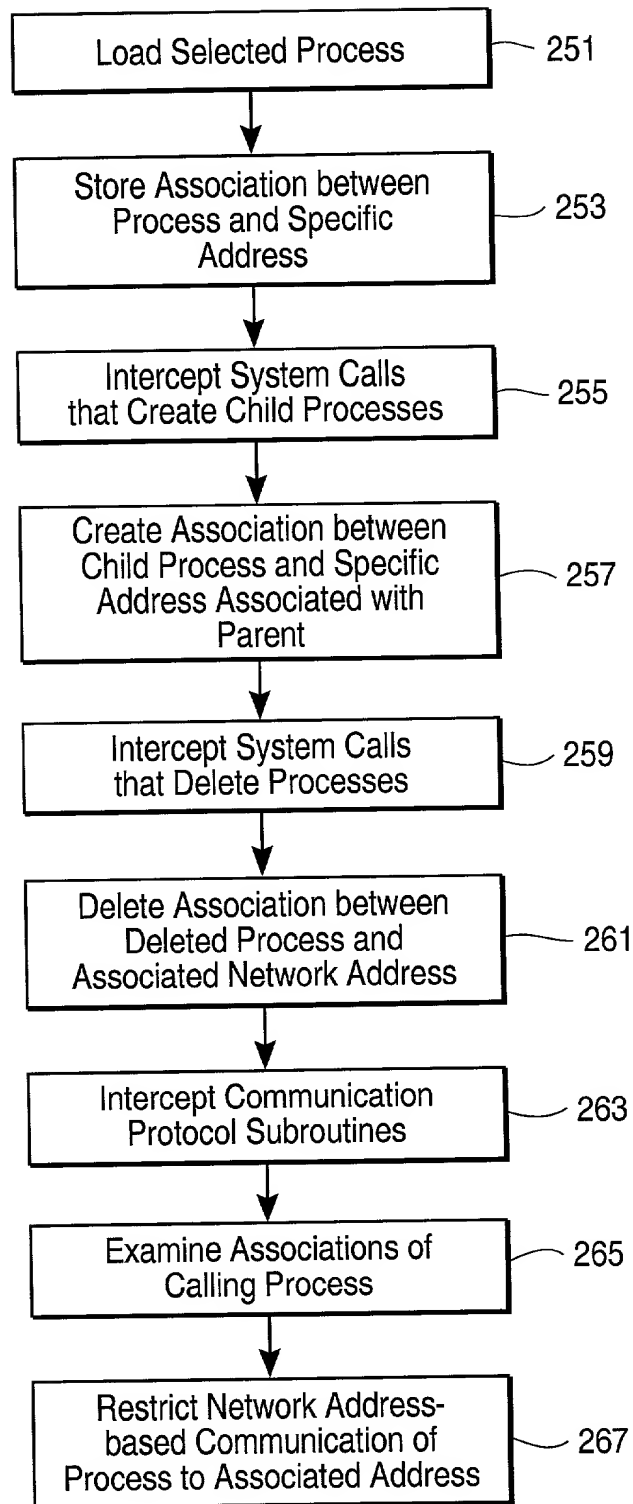
FIG. 3A

FIG. 3B

0010/PTO Rev. 6/95	U.S. Department of Commerce Patent and Trademark Office	Attorney Docket Number	**4464**
		First Named Inventor	**Pawan Goyal**
DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION		*COMPLETE IF KNOWN*	
		Application Number	**not yet known**
		Filing Date	**not yet known**
		Group Art Unit	**not yet known**
[X] Declaration Submitted with Initial Filing OR [ ] Declaration Submitted after Initial Filing		Examiner Name	**not yet known**

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

RESTRICTING COMMUNICATION OF SELECTED PROCESSES TO A SET OF SPECIFIC NETWORK ADDRESSES

the specification of which is attached hereto.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37 Code of Federal Regulations. § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code § 119 (a)-(d) or § 385(b) of any foreign application(s) for patent or inventor's certificate, or § 365 (a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached? YES	NO
			[ ]	[ ]	[ ]
			[ ]	[ ]	[ ]
			[ ]	[ ]	[ ]
			[ ]	[ ]	[ ]
			[ ]	[ ]	[ ]

[ ] Additional foreign application numbers are listed on a supplemental priority sheet attached hereto:

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below.

Application Number(s)	Filing Date (MM/DD/YYYY)	[ ] Additional provisional application numbers are listed on a supplemental sheet attached hereto.

DECLARATION	Page 2

I hereby claim the benefit under Title 35, United States Code § 120 of any United States application(s), or § 365(c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of Title 35, United States Code § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. Parent Application Number	PCT Parent Number	Parent Filing Date (MM/DD/YYYY)	Parent Patent Number (*if applicable*)

[ ] Additional U.S. or PCT international application numbers are listed on a supplemental priority sheet attached hereto.

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Name	Registration Number	Name	Registration Number
Robert R. Sachs	42,120	Albert C. Smith	20,355
Laura A. Majerus	33,417	Renee DuBord	42,500

[ ] Additional attorney(s) and/or agent(s) named on a supplemental sheet attached hereto.

Please direct all correspondence to:

Robert R. Sachs
Fenwick & West LLP
Two Palo Alto Square
Palo Alto, CA 94306
U.S.A.

Telephone	(650) 858-7110		Fax	(650) 494-1417

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of Sole or First Inventor:	[ ] A petition has been filed for this unsigned inventor					
Given Name	Pawan	Middle Initial		Family Name	Goyal	Suffix e.g. Jr.
Inventor's Signature	*[signature]*			Date	2/11/00	
Residence: City	Mountain View	State	CA	Country	U.S.A.	Citizenship	India
Mailing Address	777 W. Middlefield Road, #83					
Mailing Address						
City	Mountain View	State	CA	Zip	94043	Country	U.S.A.

[ ] Additional inventors are being named on supplemental sheet(s) attached hereto

21816/04464/DOCS/998963.1